An analysis of congestion controls in centralized control systems

Eugen Dedu, Grégory Bise, Julien Bourgeois

An analysis of congestion controls in centralized control systems

Eugen Dedu, Grégory Bise, Julien Bourgeois
FEMTO-ST Institute, DISC department
Numérica, Cours Leprince-Ringuet, BP 21126, 25201 Montbéliard, France
E-mail: {FirstName.LastName}@femto-st.fr

***Abstract*—This article studies different congestion control methods applied to a centralized control system, consisting in several sensors/actuators and one controller. Sensors/actuators are linked to the controller through an IP network. Depending on the data exchanged, the network can be congested. In such case, the congestion control used by data exchange becomes important. We evaluate four congestion control methods used by three classical transport protocols, UDP, TCP and DCCP. This evaluation uses ns2 network simulator. Results on a centralised control system show that TCP and DCCP offer a good trade-off on reliability vs. throughput, whereas UDP has best results provided that the network is well configured.**

***Index Terms*—congestion control, transport protocols, control systems, real-time systems.**

I. Introduction

Control systems are devices which manage the behaviour of systems. Closed-loop control systems consist of sensor(s) which gather data from the environment, actuator(s) which act on the environment, and controller(s) which receive data from sensor(s) and give commands to actuator(s). In our case, we are interested in Distributed Intelligent MEMS (DiMEMS) systems [1] which comprise all these elements but at a micro-scale. The Smart Blocks[1] and the Claytronics[2] projects are good examples of DiMEMS systems.

Control systems can be as simple as a sensor, an actuator and a controller. They can be centralized, consisting of several sensors and actuators but a single controller. They are distributed if there are several controllers too.

A networked control system (NCS) is a control system where the components are connected together through a network, and which has real-time constraints. The network is shared among the components either because it allows to reduce system costs or because the interconnection is wireless. NCSs have therefore a broad range of applications such as mobile robotic systems, ground-based like in RoboCup or unmanned aerial vehicle (UAV)[2].

Having real-time constraints means that information generated has deadlines to arrive, so the time when the information arrives to destination is important. They can be divided in soft, firm and hard [3]. In soft systems a deadline miss means that the information has a reduced usefulness, such as in sensor networks. In firm systems a miss means its information is useless, but infrequent misses are tolerable, such as in a live video/audio conversation. In hard systems, deadline misses generate system failure, such as in most robotic systems.

The network is shared among the components. Designing the network capacity so that congestion never appears is not always feasible, for example when data generated is not known in advance or when network has no fixed bandwidth such as the wireless networks subject to interference. So congestion can occur if data transmitted exceeds in size the network capacity. Congestion control is a mechanism to regulate sending rate so that data is sent at the fastest rate that still avoids congestion. This can be thought as network resource optimisation. When congestion occurs in network, data is lost. It is therefore important to analyse congestion control impact on data transmission.

In NCS research, data transfer is generally treated with low-level protocols, such as Ethernet and TDMA. Very few papers deal with congestion control in IP-based systems, yet this is a known challenge: "These challenges [for NCS] involve the optimization of performance in the face of constraints on communication bandwidth, congestion, and contention for communication resources, delay, jitter [...]" and "When communication channels in a data network are shared resources among multiple user nodes, network congestion and contention for bandwidth pose challenges for control implementations in which there are hard real-time requirements" [4]. [5] for example notices that different types of data exist in an NCS, hence different protocols or different priorities for packets can be used. In this context, [6] shows promising results for packet prioritisation, results which are even more appropriate to control systems since the network is under the control of one organization.

The contribution of this article is to discuss the interest of having congestion control in NCS. We analyze simulation results of classical congestion controls used above IP-based networks.

II. Related works

A. Networked Control of Systems

Networked Control of Systems (NCS) lies at the intersection of control and communication theories and it is therefore

This work was supported by the Smart Blocks NRA (French National Research Agency) project (ANR-2011-BS03-005).

[1] http://smartblocks.univ-fcomte.fr

[2] http://www.cs.cmu.edu/~claytronics

relevant to compare this study to NCS ones. A NCS has four main characteristics [7], [8] that have to be taken into account in order to control the whole system, bandwidth-limited communication channels, sampling and delay, packet dropout and system architecture. If the first three characteristics match our concerns which is network transmission, the last one only deals with the architecture of the system i.e. centralization vs decentralization of the control. Nevertheless, studies and modeling of quantization effect [9], [10], packet drop out [11] or consensus problems [12] are important and must be taken into account together with the congestion control.

B. Decentralized and distributed control

The topic of decentralized and distributed control of systems has been active for many years [13]. Decentralized control limits its study to special systems like spatially invariant ones [14] or nested, chained or symmetric systems [15], whereas distributed control makes less assumptions about the system [16]. Linear matrix inequalities (LMI) can be used either for decentralized [17] or for distributed systems [18]. LMI has proven to be efficient when dealing with linear systems whose physical topology is represented by an arbitrary graph [19] but these results have not been extended to the non-linear systems. It has also to be noticed that these methods propose only to control large distributed system but do not help to program it. If some problems require algorithmic solutions, these methods won't be used.

III. Background

A. Congestion control

The IP protocol, used also by Internet, does no attempt to optimise network usage when sending data, because it does not take into account availability of network resources. Network optimisation use has been leaved on/to the upper transport protocol. The classical transport protocols on Internet are TCP [20] and UDP [21], and recently DCCP was standardised too [22]. The way resource usage is taken into account is through a congestion control. Among the transport protocols presented, UDP has *no congestion control* with no data reliability, TCP has a *window-based congestion control* with 100% data reliability through retransmission (all data arrives to destination), and DCCP allows to choose a congestion control among currently two standardised controls: TCP-like [23], *similar to TCP*, and TFRC [24], an *equation-based control*, with no data reliability either.

When UDP receives a packet to be sent to network, it sends it immediately, no matter the network state. This intuitively leads to the highest sending rate, with the drawback that packets can be lost if the network is congested.

When TCP or DCCP receives a packet to be sent to network, it checks whether network is able to transport it. If yes, it is sent immediately, otherwise it is stored in a temporary buffer until network conditions allow to send it. The role of congestion control is to take care of the timings when packets are sent to the network.

TCP congestion control works as follows. At the beginning of a transmission, sender uses a slow start phase, where data sending rate increases exponentially. When data rate exceeds network bandwidth, packets are lost and sender enters to congestion avoidance phase, a network-friendly phase where data sending rate increases linearly; this is the phase used most of the time for long enough transmissions. Upon a lost packet detection, sender sends it immediately, known as fast retransmit phase, and enters a temporary fast recovery phase where it sends data at a smaller rate until it recovers from the loss, at which time it reduces by two the data rate and goes back to the congestion avoidance phase. This congestion control is used by DCCP/TCP-like too.

On the contrary, TFRC uses a sending rate equation whose parameters are filled with transmission parameters, most notable with the number of lost packets. Each RTT (Round Trip Time, the time for a packet go and back), the equation parameters are updated and data sending rate changes. This leads to a smoother changing in sending data rate, which is a useful property for some applications.

B. Simulators

Control and network research communities usually use simulators to validate algorithms. Two such simulators are TrueTime and ns2, respectively.

TrueTime is based on Matlab/Simulink and targets real-time and embedded systems. The simulation model is based on three blocs:

- TrueTime kernel, for logical controls and data processing;
- TrueTime network, for packet exchange in a local network such as Ethernet, CAN, TDMA and FDMA;
- a controller.

A fourth optional block can be used, TrueTime battery. This simulator focuses on low-level simulations. Instead, we are interested in high-level protocol simulations, where congestion control is implemented.

Network simulator version 2 (ns2) is usually used in the network research community. It provides low-level and high-level protocols. Several transport protocols are supported, and for some time we even maintained the DCCP implementation, used in the current article, in the simulator[3]. The accuracy of results for low-level protocols is low, but for high-level protocols, as in our case, it is high.

Given the support for various congestion controls, we choose ns2 for our simulations.

IV. Congestion control analysis

A. Network topology used

The network used for carrying out tests is presented in Fig. 1. Three agents are connected through a router to a controller. Agents are both sensors and actuators. The sensor sampling period is 50ms; they gather information about the environment and send to the controller sensor data through the form of a packet of 1kB (1024 bytes) of application data

[3]http://eugen.dedu.free.fr/ns2

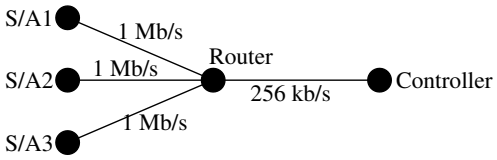

Figure 1. Network used for simulations.

each 50ms. Note that this gives slightly different packet sizes, since UDP, TCP and DCCP headers have different sizes. The first sensor starts sending at t=0, the second at t=100ms and the third at t=200ms. Upon reception of a sensor data (sensor packet), the controller answers by sending it an order through the form of a packet of 200 bytes. The topology is set so that there is a small congestion on router-controller link, whose bandwidth is a bit smaller than total sent data. The reason for this choice is that if a network is not congested, a congestion control brings of course no benefit, whereas we are interested to know if congestion control is useful.

The router uses a DropTail queue management, which means that an incoming packet is rejected if and only if the buffer is full. The buffer size is set to 50 packets, the default value on the simulator.

Each simulation is run for 360 seconds. The total number of packets generated by each sensor is 20 packets/sec * 360 sec = 7200 packets (in reality, between 7195 and 7199, since some sensors start a bit later).

Note that sending and receiving are done on different "cables", so they do not share the 256kb/s link. As controller answers are small, they do not provoke congestion, hence we are not interested by controller answers.

The figure accompanying each method shows the delays of each packet generated by each sensor, with the time on x-axis. A 0 delay means the packet is lost.

B. Results

The layer which takes care of network congestion and is responsible for packet delays is transport layer. We tested three transport protocols, the classical TCP and UDP, and a relatively new standardised protocol, DCCP. The latter can choose its congestion control, and we tested the two currently standardised: TCP-like and TFRC.

Some applications in control systems are interested by delay (time for the sensor data to arrive to controller), others by losses. On some applications values are cumulative, i.e. the value of one parameter at one time removes the need to know the value of the parameter before (e.g. the current temperature of a system or the current position of an object), which means that losses are tolerated. On others, such as videoconferencing, the delay is an important parameter, while the reliability is important for audio and not so important for video. Therefore, we have to analyse both delays and losses in simulations.

1) For UDP protocol: Fig. 2 shows that the shapes of the delays of the three sensor data packets are very different: starting from second 2, one sensor has a constant delay of 1.5s, another one a varying delay from 0s to 1.5s, and a third

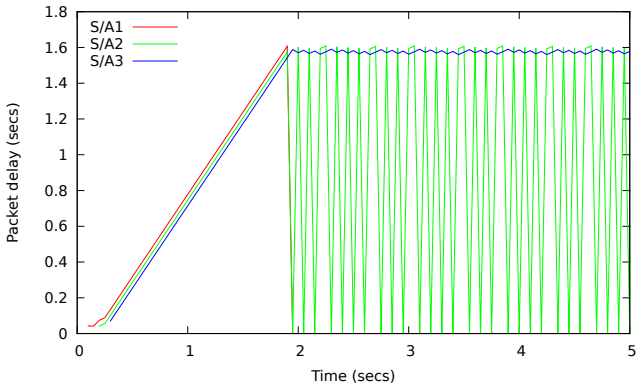

Figure 2. Results for UDP protocol for the first 5 seconds.

one has 0 delay, meaning that all its packets are lost. What happens is a timing issue: each time a packet from the latter sensor arrives at the router, its buffer is full and the packet gets rejected. Such synchronisation is a known characteristic of a DropTail queuing discipline, as set at the router, exacerbated by the fact that UDP does not use a congestion control.

10996 packets are lost, and all of them are lost by the network. No packet is retransmitted, according to UDP specification [21]. In the figure, packets start to be lost at time 1.95s. This agrees with theoretical results: During that time, about 1.95 s * 20 packets/s * 3 sensors = 120 packets have been sent, and 1.95 s * 32 = 64 packets have been forwarded by the router. This gives a difference of 56 packets, approximately equal to the buffer size on the router, set to 50 packets, as written above.

The highest delay is 1.61s, while the average delay for received packets is 1.35s. The highest delay appears for a packet which arrives at the router while its buffer is full less one packet size, hence it takes the greatest time to be processed by the router. This explanation on the highest delay stands for the whole this section, for all the protocols below.

2) For TCP protocol: Fig. 3 presents the results. The figure shows that the shapes of delays are similar, which is not surprising given that TCP uses a fair congestion control. This saw teeth-like curve is also classical for throughput of TCP protocol. At the beginning of the transmission, TCP increases very fast the data rate, and as a result the router buffer fills up and delay increases. Afterwards, TCP enters a loop where it increases slowly the data rate until a loss is detected, and reduces it by two when it arrives [20], which leads to similar changing in packet delay.

63 packets are lost by network, and according to TCP specification they have all been retransmitted and eventually received. 10382 packets are lost on the sensor itself, without entering the network. The explanation is that the built-in congestion control of TCP delays packets until network resources are available, and as a consequence TCP buffer fills up and eventually simply discards packets coming from the sensor.

The highest delay is 1.68s and the average is 1.38s.

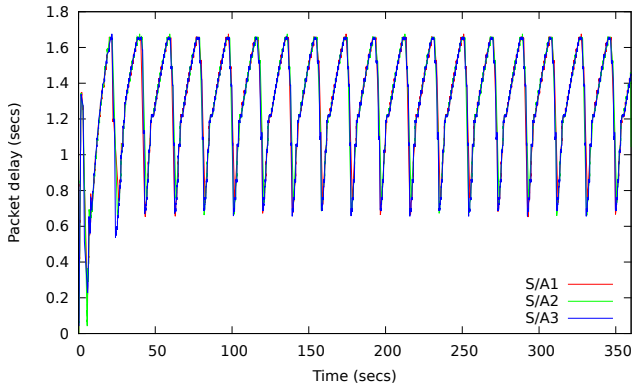

Figure 3. Results for TCP protocol for the whole simulation.

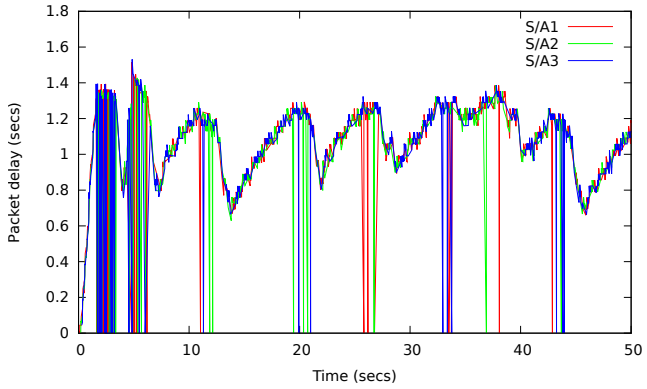

Figure 4. Results for DCCP/TCP-like protocol for the first 50 seconds.

3) For DCCP/TCP-like protocol: Fig. 4 presents the results. As for TCP, it shows that the shapes of the transfer time of the three sensor data packets are similar. This is not a surprising result, since TCP-like congestion control uses the same law for sending rate as TCP.

339 packets are lost by the network, which is a bit greater than for TCP. None of them has been retransmitted, since DCCP does not retransmit them (even if the exact number of each lost packets is known by the sender). 7021 packets are lost on the sensor for the same reason as for TCP.

The highest delay is 1.53s and the average is 1.09s.

4) For DCCP/TFRC protocol: Fig. 5 presents the results. Like TCP, TFRC is a fair congestion control, so the shapes of the transfer time of the three sensor data packets are similar.

174 packets are lost by the network, which is a bit greater than for TCP. None of them has been retransmitted, since as written above DCCP does not retransmit them. 9873 packets are lost on the sensor for the same reason as for TCP.

The highest delay is 1.63s, and the average is 1.41s.

C. Discussion

Table I presents a numerical overview of the simulation results (the delay columns show results for all the sensors). Several conclusions can be inferred.

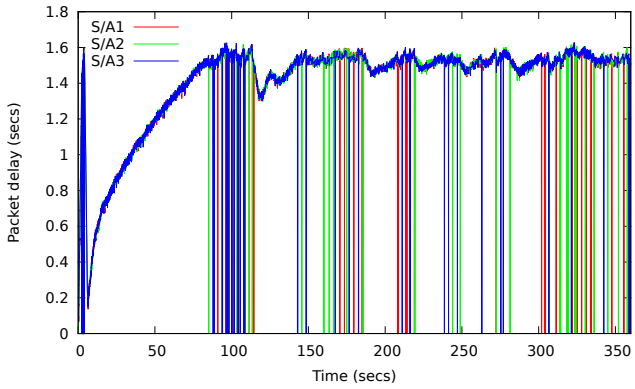

Figure 5. Results for DCCP/TFRC protocol for the whole simulation.

Using UDP means that some sensors can be almost completely muted by synchronisation issues, as given by the first sensor. Also, UDP has by far the most losses on the network, while the protocols with congestion controls avoid it and lose almost all their packets on the sensor itself. DCCP/TCP-like has the highest number of packets received, with the three others a bit behind it.

The highest delay is more or less comparable, since it is obtained when the router buffer is filled, and this case appears for all the protocols. Note that packets can wait on sensor buffer too, but this buffer is drained much faster since the link is much greater than the bottleneck link (1 Mb/s vs. 256 kb/s). On the other hand, the average delay is very favorable to DCCP/TCP-like (1.09 secs).

These results are not surprising, as explained in the following.

First, if a network is *not* congested, a congestion control brings of course no benefit. So in the following we are interested what happens when data size is greater than network capacity.

Internet is a network where flows appear and disappear generally irregularly, because user behaviour for creating flows is unpredictable and sometimes interactive. In contrast, control systems use sensors which usually send data regularly. Such regularity sometimes appears in Internet too, for example for constant video transmission such as television broadcasting. For such cases, where data is sent at a regular pace, the sending rate smoothing done by congestion control is of no help. It should also be noted that when there are several destinations with various bandwidths, congestion control can avoid useless packet losses, situation known as network collapse, an example of which is given in [25]. This is however not the case in a centralized control system which is the focus of the current article.

The delay is affected too by the congestion control. A congestion control usually delays packet sending on sender, waiting for network availability or for acknowledgement reception (also known as TCP pacing), which increases delay on sender. On the other hand, a congestion control keeps router buffers smaller, hence reducing packet delay on routers.

Protocol	Sensor	Packets				Delay	
		generated	lost on sensor	lost on network	received	highest	average
UDP	1	7199	0	7163	36	1.61	1.35
	2	7197	0	3432	3765		
	3	7195	0	0	7195		
TCP	1	7199	4425	0 (26 retr)	2774	1.68	1.38
	2	7197	4380	0 (26 retr)	2817		
	3	7195	1577	0 (11 retr)	5618		
DCCP/TCP-like	1	7199	2276	115	4808	1.53	1.09
	2	7197	2510	108	4579		
	3	7195	2235	124	4836		
DCCP/TFRC	1	7199	3496	60	3643	1.63	1.41
	2	7197	3184	54	3959		
	3	7195	3193	60	3942		

Table I
NUMERICAL RESULTS FOR THE FOUR CONGESTION CONTROLS, SAMPLING IS 50MS AND SIMULATION RUNS FOR 360 SECS.

UDP too has a limitation, because in conjunction with the simple DropTail queuing mechanism it leads to global synchronisation, preventing some sensors to send their data.

Given the discussion above, it turns out that more analysis is needed to infer what is the best congestion control. What is certain is that, in order to avoid synchronisation issues, UDP must be used together with a more sophisticated queuing mechanism, such as RED [26], [27], which rejects packets randomly before the queue is full.

V. CONCLUSIONS

This article presented a comparison among several congestion controls used by currently-used transport protocols for a centralized control system, based on simulation.

Results show that UDP in conjunction with a simple queue management mechanism has some crucial issues, but a more sophisticated mechanism should solve them. Aside that, given that the network is slightly congestion, no protocol or congestion control is definitely better than the others in terms of number of packets received. This is because in a control system data is generated at regular intervals, and the smoothing effect of congestion control has no effect. On the other hand, in terms of delay, DCCP/TCP-like gives best results. What parameter is more important depends on the control system.

Future works will use a different queue mechanism for the router, and most importantly will use congestion control to adapt sensor sending rate to network conditions, such as reducing sampling rate, which allows to have a control over data losses. We will also focus on distributed control systems, consisting of several controllers connected through a potentially complex network.

REFERENCES

[1] J. Bourgeois and S. Goldstein, "Distributed intelligent mems: Progresses and perspectives," in *ICT Innovations 2011*, ser. Advances in Intelligent and Soft Computing, L. Kocarev, Ed. Springer Berlin / Heidelberg, 2012, vol. 150, pp. 15–25.

[2] S. Chaumette, R. Laplace, C. Mazel, and R. Mirault, "Scual, swarm of communicating uavs at labri: An open uavnet testbed," in *14th International Symposium on Wireless Personal Multimedia Communications, WPMC 2011, Brest, France, October 3-7*, 2011, pp. 1–5.

[3] P. A. Laplante and S. J. Ovaska, *Real-Time Systems Design and Analysis: Tools for the Practitioner*, 4th ed. Wiley, 2011.

[4] J. Baillieul and P. J. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 9–28, Jan. 2007.

[5] R. Rajesh, M. Lakshmanan, and V. Noor Mohammed, "Implementation of networked control systems using TCP/IP," *International Journal of Computer Applications*, vol. 18, no. 2, pp. 1–5, Mar. 2011.

[6] E. Dedu and E. Lochin, "A study on the benefit of TCP packet prioritisation," in *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, ser. 17, D. E. Baz, F. Spies, and T. Gross, Eds. Weimar, Germany: IEEE, Feb. 2009, pp. 161–166.

[7] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138 –162, jan. 2007.

[8] T. Yang, "Networked control system: a brief survey," *Control Theory and Applications, IEE Proceedings* -, vol. 153, no. 4, pp. 403 – 412, july 2006.

[9] W. P. M. H. Heemels, D. Nesic, A. R. Teel, and N. van de Wouw, "Networked and quantized control systems with communication delays," in *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009*, 2009, pp. 7929–7935.

[10] D. Nesic and D. Liberzon, "A unified framework for design and analysis of networked and quantized control systems," *IEEE Transactions on Automatic Control*, vol. 54, pp. 732–747, 2009.

[11] J. Wu and T. Chen, "Design of networked control systems with packet dropouts," *Automatic Control, IEEE Transactions on*, vol. 52, no. 7, pp. 1314 –1319, july 2007.

[12] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215 –233, jan. 2007.

[13] J. Sandell, N., P. Varaiya, M. Athans, and M. Safonov, "Survey of decentralized control methods for large scale systems," *Automatic Control, IEEE Transactions on*, vol. 23, no. 2, pp. 108 – 128, apr 1978.

[14] B. Bamieh, F. Paganini, and M. Dahleh, "Distributed control of spatially invariant systems," *Automatic Control, IEEE Transactions on*, vol. 47, no. 7, pp. 1091 –1107, jul 2002.

[15] P. Voulgaris, "A convex characterization of classes of problems in control with specific interaction and communication structures," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 4, 2001, pp. 3128 –3133.

[16] P. Massioni and M. Verhaegen, "Distributed control for identical dynamically coupled systems: A decomposition approach," *Automatic Control, IEEE Transactions on*, vol. 54, no. 1, pp. 124 –135, jan. 2009.

[17] F. Borrelli, T. Keviczky, G. J. Balas, G. E. Stewart, K. Fregene, and D. N. Godbole, "Hybrid decentralized control of large scale systems," in *HSCC*, ser. Lecture Notes in Computer Science, vol. 3414. Springer, 2005, pp. 168–183.

[18] R. D'Andrea and G. Dullerud, "Distributed control design for spatially interconnected systems," *Automatic Control, IEEE Transactions on*, vol. 48, no. 9, pp. 1478 – 1495, sept. 2003.

[19] C. Langbort, R. S. Chandra, and R. D'Andrea, "Distributed control design for systems interconnected over an arbitrary graph," *IEEE Trans. Automatic Control*, 2004.

[20] J. Postel, "Transmission control protocol," IETF standard, Sep. 1991, RFC 793.

[21] ——, "User Datagram Protocol," IETF standard, Aug. 1980, RFC 768.
[22] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," IETF standard, Mar. 2006, RFC 4340.
[23] S. Floyd and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) congestion control ID 2: TCP-like congestion control," IETF standard, Mar. 2006, RFC 4341.
[24] S. Floyd, E. Kohler, and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) congestion control ID 3: TCP-friendly rate control (TFRC)," IETF standard, Mar. 2006, RFC 4342.
[25] S. Floyd, "Congestion control principles," Sep. 2000, RFC 2914.
[26] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
[27] B. Braden, D. Clark, J. Crowcroft *et al.*, "Recommandations on queue management and congestion avoidance in the internet," IETF standard, Apr. 1998, RFC 2309.